Exhibit 10

	https://developer.android.com/media/media3/exoplayer; https://developer.android.com/media/media3/exoplayer/network-stacks. https://source.android.com/docs/core/media/media-modules For example, Android MediaPlayer and/or Media3 and/or ExoPlayer comprise APIs that provide functionality to download media. This functionality is described in the links above and at https://developer.android.com/media/media3/session/player. As another example, Android framework comprises APIs that provide functionality to download media. As an example, Android framework comprises MediaSession APIs, Media APIs, and numerous additional APIs comprising Android APIs for media download.
1[e] a media service manager prompted by the second call, to manage network data transfers for the media object by interfacing with the network stack to retrieve the media object associated with the network resource identifier via the wireless modem and the wireless network; and	Samsung mobile devices, such as phones, tablets, and wearables, comprise "a media service manager prompted by the second call, to manage network data transfers for the media object by interfacing with the network stack to retrieve the media object associated with the network resource identifier via the wireless modem and the wireless network." For example, applications running on Samsung devices use, for example, the MediaPlayer and Media3 functionality, to manage network data transfers. The MediaPlayer and/or Media3 and/or ExoPlayer APIs allows applications to make data transfer requests for a media object associated with a network identifier supplied by the calling application. MediaPlayer APIs and/or Media3 APIs and/or ExoPlayer APIs in Android multimedia framework include a media service manager prompted to manage network data transfers for the media object by interfacing with the network stack to retrieve it from the wireless network. As another non-limiting example, a second API can correspond to the Android API. As another non-limiting example, Android supports one or more download managers or download services, which manage network data transfers for media objects by interfacing with the network stack to retrieve the media object associated with the network resource identifier via the network. Applications can use Android download manager API functionality by calling the Android API to make a media download (or upload) associated with a network resource identifier supplied by the calling application. The application can set allowed network types, set title, set description, set whether the app is visible in the downloads user interface, set reference identifiers, and set broadcast receiver to notify of completed downloads. A registered application calling the Android API is notified by Android API of information regarding the media download. For example, Android API can be queried to

respective data packet flow, and to reconcile wireless network data usage for each of the plurality of device applications to track an aggregate wireless network data usage attributable to each of the plurality of device applications via both the first network stack API and the second API.	For example, Samsung devices classify and measure data usage on a per-application basis and use the Android operating system data usage monitoring and tracking functionality to associate wireless data usage for media object network data transfers with the requesting applications, associate respective data packet flows via the first network stack API with the respective corresponding opening device applications, and to reconcile wireless network data usage for a plurality of applications to track an aggregate wireless network data usage attributable to each device application via both the first network stack API and the second API. For example, Android supports per-application data usage monitoring and tracking functionality, socket tagging functionality, and functionality to set limits and react when limits are reached, restrict background data traffic for an application, track network traffic on a per-socket basis for every application using the unique UID of the owning application, tagging the UID responsible for the data transfer and tagging characterizations of the traffic into one or more sub-categories, including but not limited to tracking usage attributable to the application via the first network stack API and the second API. For example, Android has functionality using application level tags to attribute ownership of network data transfer to requesting application UID in connection with download manager or media streaming framework or multimedia framework or other network data transfer service managers in Android framework or through Android APIs. For example, Android has functionality that allows media apps, download managers, media download services, and other apps to expose information to other processes such as the Android framework and other apps, facilitating these functionalities. These non-limiting examples are described below:

Monitor data usage

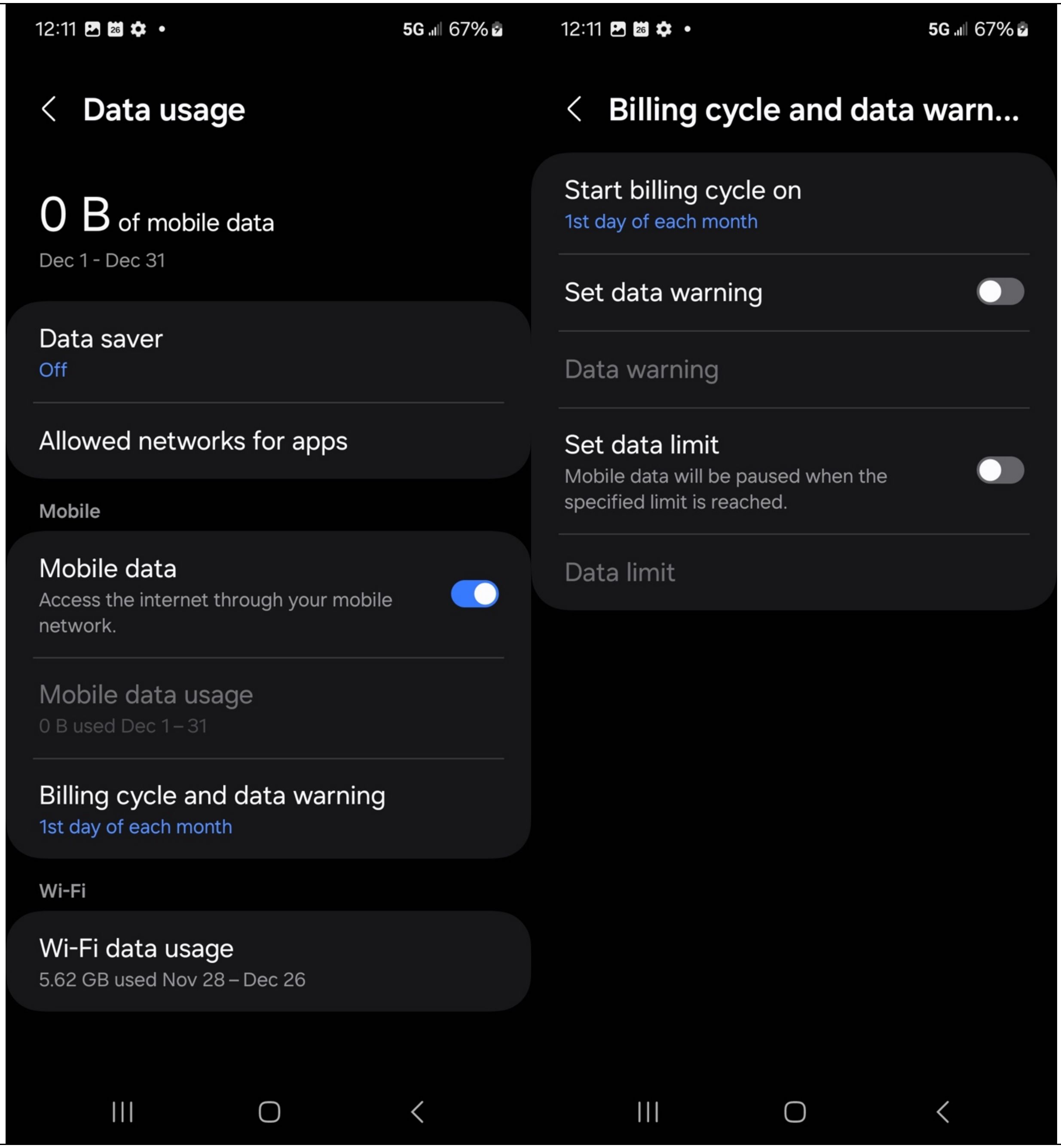
Do you know where your data is? Track your data usage to make sure you never run out.

Navigate to and open **Settings**, then tap **Connections**, and then tap **Data usage** to view your Mobile data usage or Wi-Fi data usage.

The options may be slightly different depending on what carrier you have, but here are the general settings you can adjust from this page.

- **Data saver:** Helps cut down your data usage by preventing apps from using data in the background.
- **Mobile data:** Access the internet through your mobile network. You can turn mobile data on or off by tapping the switch.
- **Mobile data only apps:** Set apps to always use mobile data, even when your phone is connected to Wi-Fi.
- **Mobile data usage:** Get a more detailed breakdown of your data usage, such as which apps used the most data.
- **Billing cycle and data warning:** Set limits or warnings to make sure you don't overuse your data. Please see the section titled "Data billing cycle settings."
- **Wi-Fi data usage:** Get a more detailed breakdown of your Wi-Fi usage, such as which apps used Wi-Fi the most.

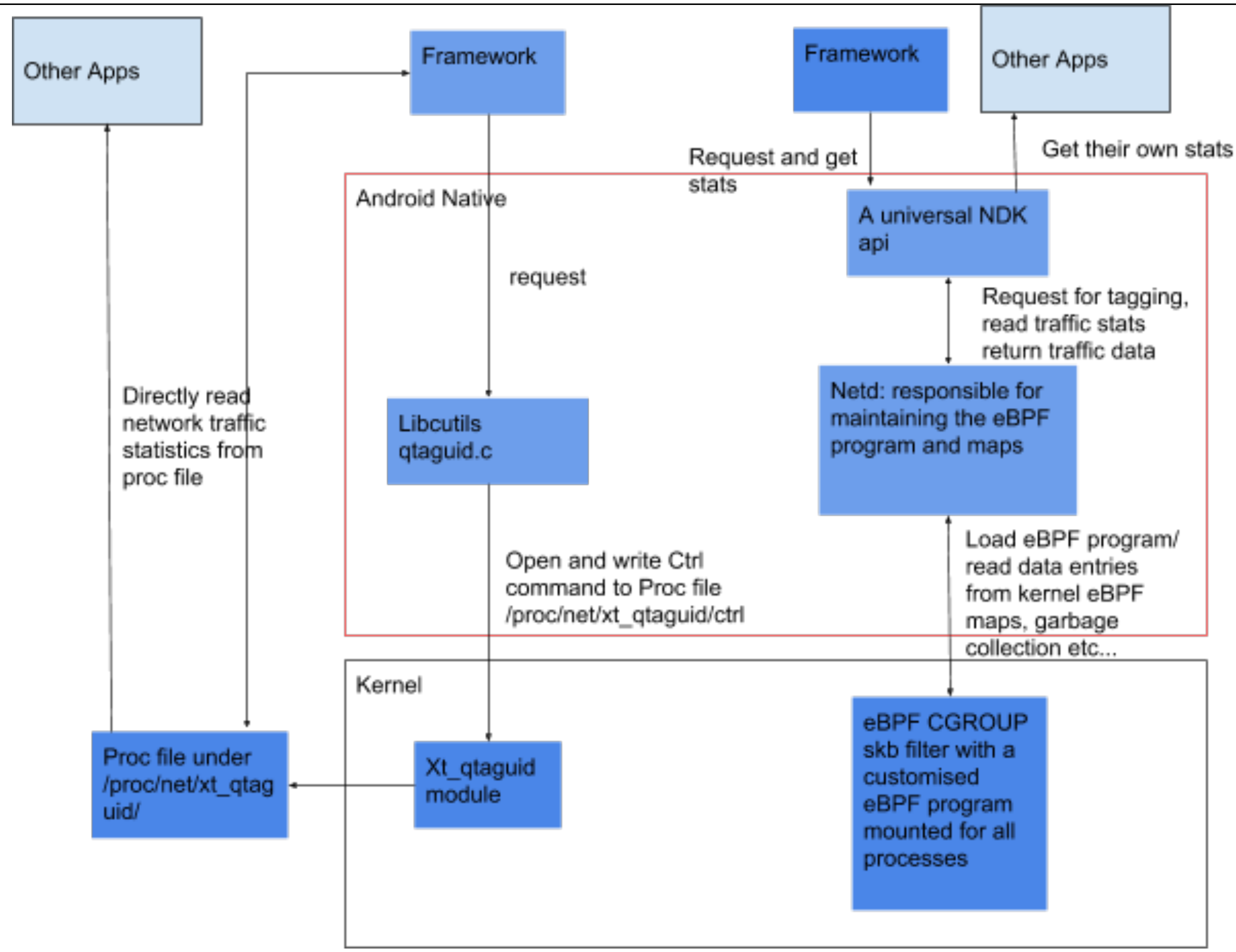
https://www.samsung.com/us/support/answer/ANS00079018/;


12:11
5G 67%
Data usage
0 B of mobile data
Dec 1 - Dec 31
Data saver
Off
Allowed networks for apps
Mobile
Mobile data
Access the internet through your mobile network.
Mobile data usage
0 B used Dec 1 – 31
Billing cycle and data warning
1st day of each month
Wi-Fi
Wi-Fi data usage
5.62 GB used Nov 28 – Dec 26
12:11
5G 67%
Billing cycle and data warn...
Start billing cycle on
1st day of each month
Set data warning
Data warning
Set data limit
Mobile data will be paused when the specified limit is reached.
Data limit

	Android Developers > Develop > Reference
	Was this helpful?
	# TrafficStats
	Added in API level 8
	Kotlin | **Java**
	`public class TrafficStats`
	`extends Object`
	java.lang.Object
	↳ android.net.TrafficStats
	Class that provides network traffic statistics. These statistics include bytes transmitted and received and network packets transmitted and received, over all interfaces, over the mobile interface, and on a per-UID basis.
	These statistics may not be available on all platforms. If the statistics are not supported by this device, `UNSUPPORTED` will be returned.
	Note that the statistics returned by this class reset and start from zero after every reboot. To access more robust historical network statistics data, use `NetworkStatsManager` instead.
	https://developer.android.com/reference/android/net/TrafficStats;

	Android Developers > Develop > Reference
	Was this helpful?
	# NetworkStats
	Added in API level 23
	Kotlin | **Java**
	`public final class NetworkStats`
	`extends Object implements AutoCloseable`
	java.lang.Object
	↳ android.app.usage.NetworkStats
	Class providing enumeration over buckets of network usage statistics. `NetworkStats` objects are returned as results to various queries in `NetworkStatsManager`.
	https://developer.android.com/reference/android/app/usage/NetworkStats;

	AOSP > Docs > Core Topics Was this helpful? **eBPF traffic monitoring** The eBPF network traffic tool uses a combination of kernel and user space implementation to monitor network usage on the device since the last device boot. It provides additional functionality such as socket tagging, separating foreground/background traffic and per-UID firewall to block apps from network access depending on phone state. The statistics gathered from the tool are stored in a kernel data structure called `eBPF maps` and the result is used by services like `NetworkStatsService` to provide persistent traffic statistics since the last boot. https://source.android.com/docs/core/data/ebpf-traffic-monitor; **Examples and source** The userspace changes are mainly in the system/netd and framework/base projects. Development is being done in AOSP, so AOSP code will always be up to date. The source is mainly located atsystem/netd/server/TrafficController*, system/netd/bpfloader, and system/netd/libbpf/. Some necessary framework changes are in framework/base/ and system/core as well. **Implementation** Starting with Android 9, Android devices running on kernel 4.9 or above and originally shipped with the P release MUST use eBPF-based network traffic monitoring accounting instead of xt_qtaguid. The new infrastructure is more flexible and more maintainable and does not require any out-of-tree kernel code. The major design differences between legacy and eBPF traffic monitoring are illustrated in Figure 1.

	 **Figure 1.** Legacy (left) and eBPF (right) traffic monitoring design differences The new trafficController design is based on per-cgroup eBPF filter as well as xt_bpf netfilter module inside the kernel. These eBPF filters are applied on the packet tx/rx when they pass through the filter. The cgroupeBPF filter is located at the transport layer and is responsible for counting the traffic against the right UID depending on the socket UID as well as userspace setting. The xt_bpf netfilter is